Dynamic Resource Allocation for a Sensor Network

Manuscript #335

Authors:

Paolo Gaudiano
Iavor Trifonov
Martin C. Martin
Eric Bonabeau

Icosystem Corporation
10 Fawcett Street
Cambridge, MA 02138, USA

{paolo,iavor,martin,eric}@icosystem.com

Corresponding Author:

Paolo Gaudiano

Icosystem Corporation
10 Fawcett Street
Cambridge, MA 02138, USA
paolo@icosystem.com
+1-617-520-1070

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JUN 2005		**00-00-2005 to 00-00-2005**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Dynamic Resource Allocation for a Sensor Network	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Icosystem Corporation,10 Fawcett Street,Cambridge,MA,02138**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**21**	

Dynamic Resource Allocation for a Sensor Network

Paolo Gaudiano, Iavor Trifonov, Martin C. Martin, Eric Bonabeau
Icosystem Corporation
10 Fawcett Street, Cambridge, MA 02138, USA
{paolo,iavor,martin,eric}@icosystem.com

1 Abstract

We describe a project that analyzes the performance of a sensor network trying to detect enemy agents in an urban environment. We simulate a network of units that can detect different enemy types using a variety of sensors. Our work aims to address key issues about networked systems, including the effectiveness of cooperative strategies, the balance of multiple constraints, and the tradeoffs between resource allocation and performance. We make use of evolutionary computing algorithms to search the space of possible configurations, carrying out parameter sweeps to test how various dimensions of the design space impact performance over a range of scenarios. Our results provide some quantitative insights into the performance of networked systems.

2 Overview

The concepts of Network-Centric Warfare (NCW) and Network-Centric Operations (NCO) have been the target of significant interest in the last few years from the defense community (Cebrowski and Gartska, 1998; Alberts and Hayes, 2003). While there is general agreement that NCO holds great promise for military applications, many significant obstacles remain before these concepts can be fully deployed.

Designing a system of networked military units presents significant challenges in terms of at least three key factors: (1) allocation of the appropriate resources; (2) integration of the various components in terms of their coordination, cooperation, command, and control in achieving a focused set of specific and cohesive goals; (3) definition of the desired rules of behavior that are scalable and robust across a variety of scenarios and complex environments.

The need to address all of these issues has lead to the high specialization of solutions and the consequent narrowing of their applicability, which is problematic in a complex and uncertain world. This problem permeates today's military at all levels, from planning and design of military platforms to the implementation and execution of specific missions.

The Wolf PAC initiative sponsored by the Naval Undersea Warfare Center (NUWC) in collaboration with the Office of Force Transformation addresses these problems by focusing on the design of distributed and adaptive systems that adequately serve the needs of Special Operation Forces operating in highly dynamic and unpredictable environments. As part of this initiative Icosystem Corporation has received support (contract N66604-05-0442) to explore the application of agent-based modeling (ABM)

and evolutionary computation (EC) as techniques for addressing the platform design and behavior analysis problems described above.

In particular, we worked on the problem of resource allocation by exploring the tradeoffs that face the design of a robust distributed adaptive operation (DAO) network system for the military. The problem is examined in the context of using a set of networked military assets to detect, identify and track enemy units that traverse an environment. Specifically, we investigate the allocation and behavioral rules of multiple heterogeneous and distributed networked assets within a complex environment and limited budgets. This allows us to demonstrate the natural application of ABM and EC to both the problem of resource allocation and the problem of behavioral modeling.

For this project, we have chosen to model a set of dispersed, networked units (*smart dust agents*) whose mission is to detect, identify, and track hostile agents in an urban environment using one or more sensor types. This simulation model is then used in an evolutionary framework to evolve and discover efficient allocation strategies of resources across friendly units and the appropriate rules for their cooperative behavior.

The most significant aspect of this work is that we take into account several of the constraints that must be considered for deployment of a networked system. In a traditional sensor network approach, one might specify a given configuration of sensors (characteristics and layout) and try to find an optimal algorithm. However, that algorithm may be suboptimal (or fail miserably) as a result of changes in the sensors, in the network they form, or in the environment. In contrast, we look for strategies that consider simultaneously the following issues:

- The types of sensors available
- The sensor accuracy
- The radius of communication between sensors
- The way in which information is shared among sensors
- The environment in which the sensors are operating

The first three of these items jointly define the cost of deploying the networked units, while the last two determine how well a given configuration will perform.

To study the tradeoffs between the cost of the resources that we employ and their overall performance as a system, we look at a set of experiments in which we manipulate three different dimensions:

1. The available budget
2. Whether or not the units cooperate
3. The number of distinct enemy types

By comparing the results across these dimensions, we found some performance patterns that hold across multiple scenarios. We summarize here some of our main findings.

- In general, cooperating units outperform non-cooperating units

- Increasing available budget always results in a performance increase at any given level of scenario complexity
- As the budget becomes unlimited, allowing for a large number of sensors with high accuracy, cooperation becomes unnecessary

Before describing our model and the simulation results, we summarize our approach.

2.1 Summary of the approach: Agent-Based Modeling

For our simulation of the smart dust network and enemy agents, we use a modeling approach known as *agent-based modeling* (ABM). In this approach, systems are represented as collections of autonomous decision-making entities, called agents. Each agent individually assesses its situation and makes decisions based upon a set of behavioral rules. Agents may execute various behaviors appropriate for the system they represent – for example, producing, consuming, or selling. At the simplest level, an agent-based model consists of a system of agents and the relationships between them. Even a simple agent-based model can exhibit complex behavioral patterns (Epstein & Axtell, 1996; Bonabeau, 2000, 2002a, 2002b) and provide valuable insight about the dynamics of the real-world system that it emulates.

The benefits of ABM over other modeling techniques can be captured in three statements: (1) agent-based modeling captures emergent phenomena; (2) agent-based modeling provides a natural description of complex systems; (3) agent-based modeling is flexible. While the ability of ABM to deal with emergent phenomena is the main driving force behind its success as a complex adaptive systems (CAS) tool, the fact that it provides a natural framework to describe complex systems combined with its flexibility makes ABM the tool of choice for the high-dimensional design space problems addressed in this project.

2.2 Evolving robust systems

The ability to model complex systems allows us to analyze and exploit their characteristics in a vastly improved manner. Experiments and various scenarios can be simulated to test solutions to our problems, while the extensibility and scalability of the ABM approach enables us to conveniently build on existing platforms.

At the same time, the ability for accurate and easy representation of complex systems opens the door to high-dimensional design spaces. These design spaces are not amenable to exhaustive search. The requirement of robust solutions that are relevant in a variety of situations and scenarios complicates the task even further. Evolutionary algorithms address this problem by allowing for a highly efficient, automated search of vast design spaces. Evolutionary algorithms (EA) enable us to encode solutions in a deterministic way and evolve optimal instances based on the principle of natural evolution.

Using ABM and EAs together allows us to explore the solutions of complex problems in the context of their natural setting. An agent-based model of the system in which a specific problem is addressed serves as a test bed for evaluating a proposed solution. The availability of an automated ABM simulation of the environment and the outcomes of the application of a particular strategy enables the use of EAs for the efficient, automated search of the problem design space. This approach requires that intervention rules are

designed as elemental building blocks that can be meaningfully recombined and adjusted through evolutionary mechanisms such as Genetic Algorithms (Goldberg, 1989; Forrest, 1993). With evolutionary algorithms, robustness can be made an integral part of the search process.

3 Model description

To address the problem of resource allocation and military asset coordination we designed a framework that consist of two main components: an agent-based model that simulates the behavior of the system at hand, and an evolutionary algorithm that uses the model as a test bed for evaluating and evolving resource allocation and behavioral (i.e., command and control) rules for networked assets. In addition, the simulation explicitly models enemy agents and the environment in which the scenario takes place.

3.1 The agent-based model

For the purpose of illustrating the applicability of agent-based modeling and evolutionary computation to the problem of military resource allocation and coordination in complex environments we consider a specific problem in a simulated environment. In particular, we look at the problem of distributing resources within a sensor network on the battlefield, and designing rules for coordination and information sharing between nodes.

The system under consideration consists of three major components: a complex environment, a set of heterogeneous 'Smart Dust'- like friendly agents, and a set of heterogeneous enemy agents. The configuration of the environment, the characteristics of the available friendly agent types, and the configuration of the enemy agents are provided to the model as input. We then simulate the detection, identification and tracking of enemy agents that traverse the environment. The model provides as its main output the cost required to select and equip the friendly agents and the performance of the system in terms of its ability to detect enemy agents.

The interaction between enemy and friendly agents is based on the detection of signals. Enemy agents emit different kinds of signals, e.g. sound, chemical, heat, vibration, etc., while the smart dust agents have a suite of on-board sensors that have the ability to detect these types of signals in order to detect and identify the source and its type.

3.1.1 The simulated environment

The environment is a simplified two-dimensional urban grid of a preset, configurable size. The map is generated using a stochastic algorithm that ensures the resulting map is randomly created, but still follows a consistent pattern that resembles the layout of a city to a degree sufficient for our purposes. Figure 1 displays a sample map of a simulated urban environment, with friendly agents randomly spread throughout the environment (displayed as white dots). Enemy units traverse the environment by following a sequence of waypoints, which are shown in Fig. 1 as red dots connected by gray lines.

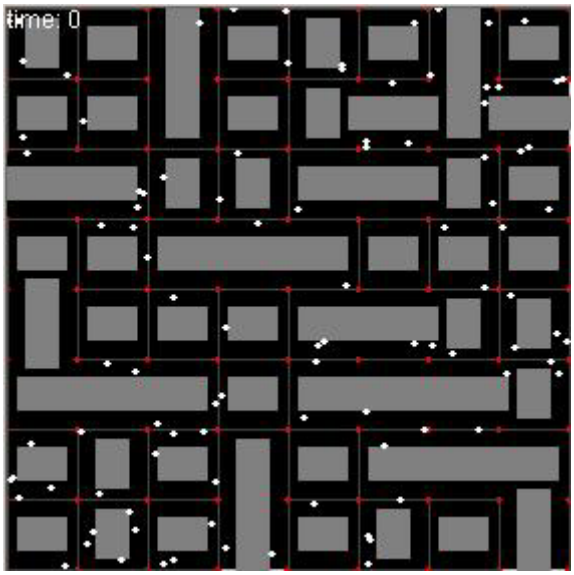

Figure 1: A sample urban map

The algorithm for generating the maps splits the area into a grid, where each cell contains a city block. The algorithm loops through all cells and creates a city block that is either horizontal or vertical. The probability for placing a vertical block is calculated by a sinusoidal function that depends on the number of the cell that the block is being placed in. This ensures local consistency with small variations, with changes from predominantly vertical to predominantly horizontal blocks on the bigger scale.

In addition, a parameter determines the probability that a block will extend to connect to its neighboring blocks in order to make continuous blocks that create more convoluted paths around the city. This parameter may be used to increase or decrease the connectivity of the maps that are being generated.

3.1.2 Enemy Agents

Enemy agents traverse the environment by selecting starting points and destinations at random. They plan their routes by following the shortest path to their destination. There may be more than one enemy agent at a time, which the network of smart dust units is trying to detect and track.

Enemy agents are differentiated by the types of signals that they emit. They emit different signals like motion, heat, chemical, etc., with different intensity, thereby requiring different types of sensory equipment for detection. The set of signals that an enemy agent emits defines its signature, which is used for identification by the smart dust agents.

For simplicity we have limited the total number of different signals to three, which provides for seven different enemy signatures (each signal type can be emitted or not emitted by an enemy, and we exclude the case that no signals at all are emitted). Each type of enemy agent is further characterized by the speed with which it moves through the environment, and by how far its signals are emanated. For instance, a single enemy soldier may emit sound only, move slowly, and its sound signal may only be detectable from a short distance. Details of the enemy agent types and characteristics are provided in a later section.

3.1.3 Smart Dust Agents

Smart dust agents perform detection, identification, and tracking of enemy agents. To perform detection smart dust agents use an array of sensors and cooperate with each other. Each sensor has the ability to detect a specific type of signal with certain accuracy. This means that the sensors are imperfect, having a probability of reporting that a signal is present when it isn't (false positive), as well as reporting that it is not present when it is (false negative).

All smart dust agents have the ability to produce a predictive indication for each of the existing enemy types and have the same radius of communication. Their heterogeneity is found in the array of available sensor types that they have on board. There is a fixed set of available sensor types (one for each existing signal type) but smart dust agents are assigned sensors at random with the goal of reaching a certain specified density for each sensor type in the population of all smart dust agents.

The density for each sensor type is provided as input for the simulation. This is a number in the [0…1] interval and denotes the percentage of all smart dust agents that should have this particular sensor type on board. This means that some agents will have all possible sensors on board, while others will have just two, one, or even no sensors at all. Nevertheless, every agent is capable of calculating a detection reading about whether a particular kind of enemy is in the vicinity or not, as mentioned earlier.

Detection readings range in the [-1…1] interval in a continuous fashion. A positive measurement indicates that an enemy agent of the particular type is detected in the area, while a negative value indicates that there is no enemy agent of that type in the vicinity. The determination of presence or absence of a signal is based on the sign of the reading, while the actual magnitude corresponds to certainty. The more positive the reading is, the longer it would take for it to be reduced down to the negative range. On the other hand, if the detection reading is slightly above zero, it can quickly be turned into the negative zone. Going from negative to positive follows an analogous process.

Information about every possible signal type is required for the unambiguous determination of whether a particular enemy type is present or not. If the information is incomplete, then it can be impossible to confirm or exclude the possibility that a particular enemy type is in the vicinity.

The ability of smart dust agents to produce detection readings even without a full set of sensors is achieved through cooperation. Smart dust agents are aware of other smart dust agents within their communications range and share information with one another. The information that is being shared between agents is on the level of sensor readings. In a way, smart dust agents that are missing a particular sensor can simply use the sensor reading from their neighbor. When more than one sensor reading for the same signal type is available, the information is aggregated to produce a more accurate reading. Figure 2 illustrates the flow of information between two neighboring smart dust agents.

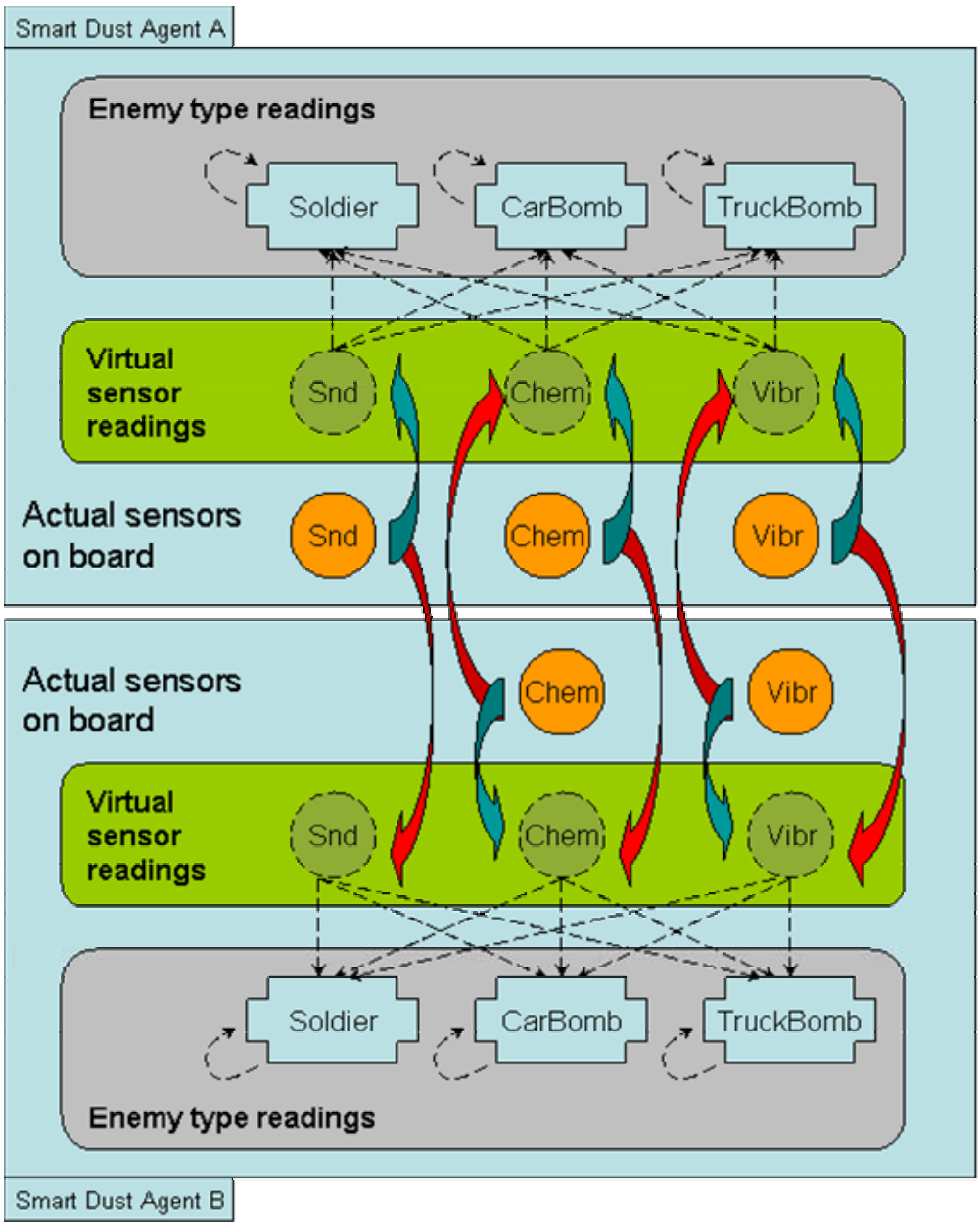

Figure 2: Information sharing between smart dust agents

As Fig 2 shows each smart dust agent has a detection reading for every enemy type. An enemy type reading indicates the certainty of the particular smart dust agent that an enemy agent of the type is present in the vicinity. This reading is a function of two sets of inputs. First, the reading takes into account its own previous value, in order to take advantage of its previous knowledge. Second, the information from the agent's own sensors is added. In other words, information is integrated both spatially and temporally.

Each sensor produces a measurement indicating whether a certain type of signal is presently being detected or not. This is used to determine whether the signature of the particular enemy type is recognized and the final reading is calculated appropriately. In order to recognize the full signature of an enemy agent however, signal information for every signal type is required. When an agent is missing a sensor type it cannot produce a

full set of signal readings. This problem can be overcome if neighboring smart dust agents share sensor information as illustrated in Fig 2.

Regardless of whether an agent has a full set of sensors, it always maintains a virtual reading for every possible sensor. When there are neighbors around, their sensor information is used to fill in the blanks and aid the existing sensors. The red arrows in Fig 2 show the flow of information between the two agents, while the blue arrows show the flow of information inside each agent.

The sharing of sensor information is the essence of cooperative behavior demonstrated in the model. In order to make any definitive decisions, the smart dust agent requires information about all signal types. When it is missing a sensor for a particular signal type, the smart dust agent needs to rely on the sensor readings of its neighbors.

Sensory information is integrated both spatially and temporally. In the spatial domain, when multiple readings are available for a given sensor type, a majority voting rule is used to determine whether a signal is present or absent. Once the multiple readings have been aggregated in this fashion, information is further integrated in time. To this end, we use a standard leaky integrator formalism. The leaky integrator balances the relative contribution of its past activation level against the contribution of the current signal (aggregated through majority voting). By modifying the relative contribution of these two terms (past activation level and new inputs), the smart dust agent can be adaptively tuned to reflect the quality of the sensory information: in the presence of a lot of noise, integration over longer time periods can reduce some of the noise; on the other hand if the sensory information is highly accurate, there is less need for temporal integration.

Finally, each smart dust agent aggregates information from all three sensor types in order to determine whether a particular enemy type is present (see Figure 2). At this stage we presume that each agent is "hard-wired" to identify three types of enemies. By adjusting the weights from each aggregated sensory node to the enemy detection nodes, the agents can discriminate between different enemy types.

3.2 The Evolutionary Algorithm

As should be apparent from the description in the last section, our model exhibits a high degree of complexity and a large number of parameters. A brute force parameter sweep to find optimal performance under all possible scenarios is infeasible. However, we can leverage a technique known as evolutionary computing to perform a guided, global search of the design space.

We employ a genetic algorithm, or GA (Goldberg, 1989), which is a type of evolutionary algorithm, to navigate the design search space for our problem and to demonstrate the effectiveness of evolutionary algorithms in combination with agent-based models for solving complex problems.

The GA is designed to minimize a cost function that captures a combination of the budget and performance. The GA manipulates the communications range of the smart dust units, the sensor characteristics (accuracy and density) and the algorithm for detection (by modifying the parameters for spatial and temporal integration of information).

There are two essential parts to a genetic algorithm that define its purpose and method of search. These are (1) the representation being used to encode candidate solutions and (2) the fitness function that is used to evaluate them. In addition, a set of genetic operators is provided to modify and expand the set of potential solutions that seem to be successful according to the fitness function evaluations. We look at each of these in turn.

3.2.1 Chromosome representation

The chromosome of the GA contains all the building blocks to a solution of the problem at hand in a form that is suitable for the genetic operators and the fitness function. The building blocks that define the characteristics and behavior of the smart dust agents are captured in four sets of genes.

First, we need to discover an appropriate range of communication between smart dust agents. This value can be captured by a single floating point number that varies between zero and a maximum allowable radius.

Second, we are looking for the appropriate density of each available sensor type across the population of smart dust agents. The density of each sensor type can be represented as a floating point number in the range [0…1]. We need one such gene for each of the three different sensor types.

Third, we include the accuracy of each sensor type. Its representation is analogous to that of the density as we need a separate gene for each available sensor type. The range of acceptable values, however, varies in the interval [0.5…1.0]. The low boundary is at 0.5 as this represents 50% accuracy. Any accuracy less than 50% results in a worse prediction than that of the flip of a coin.

The fourth gene encodes the behavior of the detection algorithm by specifying a set of weights for each agent. As described earlier, each agent accumulates information about all three possible enemy types. For each enemy type, we specify two weights: the contribution of the sensory input, and the contribution of past activation levels (which implicitly defines the decay rate of the leaky integrator). The decay rate (δ_j) and the coefficient for the sensor input (σ_j) are in the range [0…1].

As mentioned earlier, there is a cost associated with resources. In order to optimize the performance of the system within a certain budget we have to enforce a budget limit on the chromosome that is being evolved. This, in turn, requires that we define the cost of a given configuration. We calculate total cost using the following equation:

$$Cost = C\frac{1}{1-\frac{r}{2 * R_{\max}}} + S\sum^{SENSORS} d_i \frac{1}{1-a_i} \qquad \textbf{(1)}$$

In this equation, the cost function has two main terms: the first term reflects the cost of communications; the second term reflects the cost of sensor accuracy and density. The significance of the two terms can be adjusted through the weight coefficients *C* and *S*. We now explain each term of the equation:

- The cost of communications is proportional to the radius r is normalized by the maximum possible radius R_{max}, and is bounded to values between C and $2*C$.
- The cost of sensory quality is summed across all sensors, and it combines the cost of increased density d_i (which is linearly related to the number of sensors) and the cost of increased accuracy a_i. The accuracy a_i ranges between 0.5 and 1, and as it approaches perfect values, the cost quickly becomes infinite.

Every chromosome in the GA population is checked to ensure that the limited budget constraint is met. The radius, the accuracy, and the density are allowed to vary freely, but their total cost must remain within the specified maximum budget. This constraint is enforced at all times during the genetic search. If a given chromosome results in a cost that exceeds the budget, the values of d_i , a_i and r are gradually reduced until the budget constraint is satisfied.

r	$d_1 \ldots d_n$	$a_1 \ldots a_n$	$\delta_1, \sigma_1 \ldots \delta_m, \sigma_m$

Table 1: **Structure of the chromosome used for the GA**

Table 1 depicts the general form of the chromosome used in our GA. In all experiments reported below we use three sensor types (n = 3) and a variable number of enemy types for different experiments (m = 1, 2 or 3). This means that for different experiments there will be a different number of genes in the chromosome.

3.2.2 The fitness function

The fitness function is used to evaluate candidate solutions by comparing the performance that results from using a particular chromosome. Our fitness function for an individual i includes terms that measure how well the system detects, identifies and tracks enemy agents, according to the following equation:

$$F_i = \max\left\lfloor P\left(-\frac{1}{E_{TP}} + \frac{1}{1-\sqrt[4]{E_{TP}}}\right),\ N\left(-\frac{1}{E_{TN}} + \frac{1}{1-\sqrt[4]{E_{TN}}}\right)\right\rfloor \qquad \textbf{(2)}$$

In our GA we consider a lower value of F_i to correspond to better fitness. As shown in Eq. (2), the fitness function has two main terms: the first term reflects the performance in terms of the error rate in correctly identifying enemy agents when they are really there (true positives); the second term reflects performance in terms of the error rate in correctly indicating that an enemy agent is not present when it is really not there (true negatives). The coefficients P and N can be used to conveniently adjust the fitness function by giving more significance to the desired term.

Both error terms are calculated as an average over all smart dust agents over an entire simulation, which is set up with the selected communications range, accuracies, and densities. The error varies in the range [0...1], but the two different terms ensure that there is a significant benefit for small errors, as well as a big penalty for large errors.

As the error becomes large, the 1/(1-x) expression makes the sum quickly become infinite. On the other hand, the smaller the error is, the more negative the expression

becomes because of the (-1/x) term. Finally, we take the larger of the two errors in order to ensure that the chromosome is given the larger penalty of the two and to ensure that the true positive and true negative error terms do not cancel each other in the extremes.

4 Experiments and results

The main goals of this project are (1) to expose the complexity of the design space for resource allocation and behavioral coordination problems, (2) to demonstrate the effective application of agent-based modeling and evolutionary computation to this type of problem, and (3) to support the argument that cooperation between distributed, networked units can be beneficial in complex situations. In order to pursue these points, we devised a set of experiments that explore three key dimensions: the complexity of the enemy forces, the cost of the sensor network and the level of cooperation among networked units.

4.1 Enemy types

As explained earlier, our smart dust units can be equipped with sensors of three different modalities. We define different enemy agent types by the combination of signals that they emit. Having three different signal types allows for a total of seven different enemy types (assuming that each enemy type can emit one, two or all three signal types). In our experiments we define three enemy types, set up in three different configurations of increasing complexity. The three different signals that we have are sound, chemical, and vibration. Table 2 lists the three different enemy types that we use.

Enemy type	Signals emitted
Single Soldier	Sound
Car Bomb	Sound, Chemical
Truck Bomb	Sound, Chemical, Vibration

Table 2: Enemy types

The three enemy types are designed in a way that they have overlapping signal signatures, which leads to increased complexity when they are present in the world at the same time. We use these enemy types in three different combinations for our experiments. Table 3 lists the three scenarios.

Scenario	Enemy Types Enabled
1	Single Soldier
2	Single Soldier, Car Bomb
3	Single Soldier, Car Bomb, Truck Bomb

Table 3: Enemy type combination scenarios

When we run a simulation in scenario 1, the only enemy type that exists is the Single Soldier. This is our base case scenario, where we have one enemy type that emits one signal type. Scenarios 2 and 3 are of greater complexity because there are two and three enemy types respectively that exist in the world, and at the same time the additional enemy types have more signals in their signatures. The complexity comes from the fact

that we not only have to recognize more enemy types individually, but are also facing the situation of two or more different enemy types being in the same place at the same time. Because the different enemy types have overlapping signatures, it becomes harder to determine what enemy type is actually present.

4.2 Budget limits

For each scenarios we can run a GA to optimize the configuration of the smart dust agents. We perform that optimization for five different budget limits per scenario. The five budget limits are derived from five base cases for the density and accuracy of the available signal sensors. We calculate the cost when all three signal sensors have a 100% density with an accuracy of 60%, 70%, 80%, and 90%. The resulting costs (calculated from Eq.1) are the first four budget limits that we use. The last budget limit value is a large value, which represents the case of a practically unlimited budget. Table 4 lists the five budget limits.

Calculated for sensor accuracy of	Budget Limit
60%	7.5
70%	10
80%	15
90%	30
-	100000

Table 4: Budget limits

The budget limits in Table 4 are calculated for a density of 100% and the specified accuracy. However, the GA is free to vary the communication range, the density, and the accuracy in any way, as long as the combined cost does not exceed the budget limit.

4.3 Level of cooperation

In our model the level of cooperation is determined intrinsically through the weights connecting neighboring smart dust units, which are evolved by the GA. However, we decided as a fourth dimension to explore the impact of preventing cooperation altogether. Hence for each scenario we can choose to set cooperation to zero, in which case there will be no cooperation at all between units. If the cooperation is non-zero, we let the GA determine the level of cooperation.

4.4 Summary of scenarios and GA setup

By considering all possible combinations of the three search dimensions just described (enemy complexity, budget limit, cooperation level), we have a total of 30 distinct experiments. Table 5 summarizes the values that each of the key parameters can take.

Cooperation	Scenarios	Budget Limits	Total
On, Off	1,2,3	7.5,10,15,30,100000	30

Table 5: Summary of experiments

For each experiment we run the GA to find a smart dust agent configuration that has the highest performance possible within the specified budget. In each case the GA parameters are identical and their values are listed in Table 6.

Parameter	Value
Population size	50
Number of generations	30
Elite size	10
Crossover probability	0.6
Crossover points	2
Mutation probability	0.2

Table 6: GA parameters

The termination criterion for all GA runs is the number of generations. Each run is stopped after 30 generations and the best individual at this point is taken as the result for analysis. In addition, each of the 30 experiments is repeated 10 times with a set of 10 different random seeds to obtain a better estimate of the result after exactly 30 generations. All results below show the average and standard error of the mean over the 10 runs for each condition.

4.5 Results and analysis: system performance

To analyze our simulation results, we present three figures, one for each of the three enemy type scenarios listed in Table 3. We first look at the results from scenario 1 simulation runs in Fig. 3.

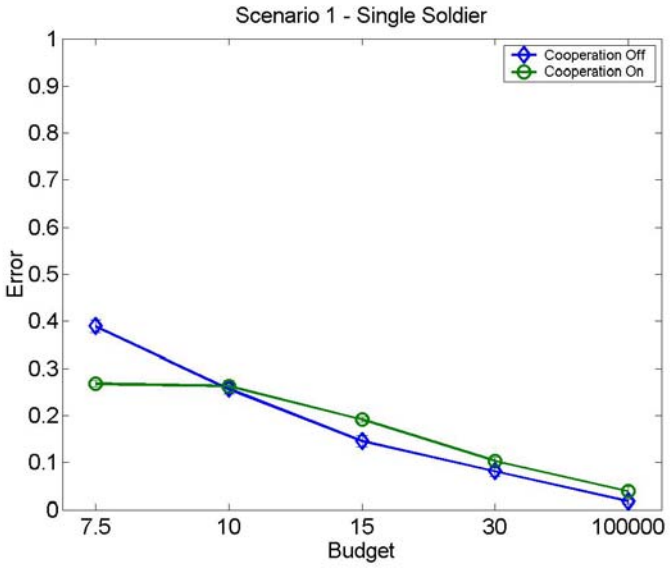

Figure 3: Performance under scenario 1

Figure 3 displays two curves – one for simulation runs with cooperation enabled (green line with circles) and one for simulation runs with cooperation disabled (blue line with diamonds). Each curve has five data points for the five different budget limits. The X-axis shows the budget limit for the particular GA run, while the Y-axis shows the error in the performance of the best individual after 30 generations. Each point is the average of ten runs with ten different random seeds. The plot includes error bars for each point, representing the standard error of the mean. Because of the simplicity of this world, the error bars in this plot are not visible as they fall entirely within the symbols drawn at each point. The error bars will become evident in subsequent results.

Scenario 1 is our base case scenario where we have only one enemy type that emits one signal. As evident from Fig 3, in both cases we have a consistent reduction in the error with the increase of the budget limit. It is interesting to note that in this scenario, cooperation seems to have a positive impact only at the lowest budget levels. At higher levels the cooperation seems to have little impact.

On the other hand, simulation runs under scenarios 2 and 3 show a consistent advantage in using cooperative agents, as show in Figures 4 and 5. For all five budget limits under scenario 2 (Fig. 4) the GA manages to find better solutions when it has the option of using cooperation. Under scenario 3, we see a significant improvement for the cooperative system for the higher budgets (Fig. 5). This suggests that for the more complex scenarios we get increased performance by using cooperation.

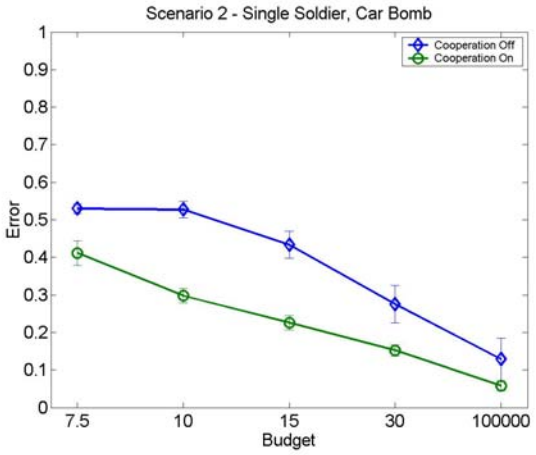

Figure 4: Performance under scenario 2

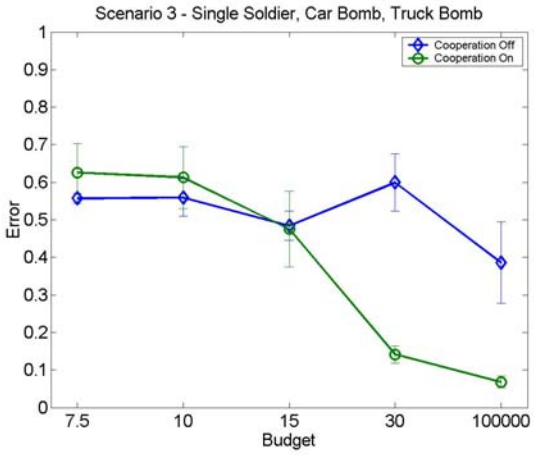

Figure 5: No city block connectivity, scenario 3

It is useful to compare the simulation runs across the three different scenarios. Figures 6 and 7 compare the performance of configurations with cooperation across the different scenarios as well as configurations without cooperation. Figure 6 clearly shows that for the two more complex scenarios the GA had a hard time finding solutions that bring the error down even for the higher budget limits when there was no cooperation between nodes. On the other hand, in Fig 7 we see that there is a consistent decrease in the error as the budget increases for all three scenarios.

It should also be noted from Figs. 6 and 7 that the curves for scenarios of increasing enemy complexity tend to be consistently on top (higher error). This suggests that our method of representing enemies does in fact result in added complexity. Nonetheless, the

GA manages to find consistently better solutions when cooperation is available, or when the budget limit is being increased.

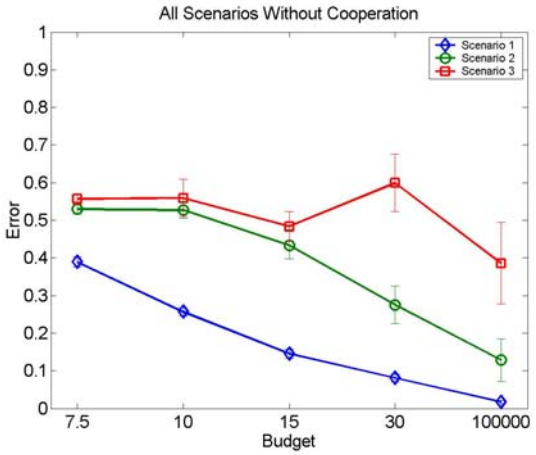

Figure 6: Performance with no cooperation

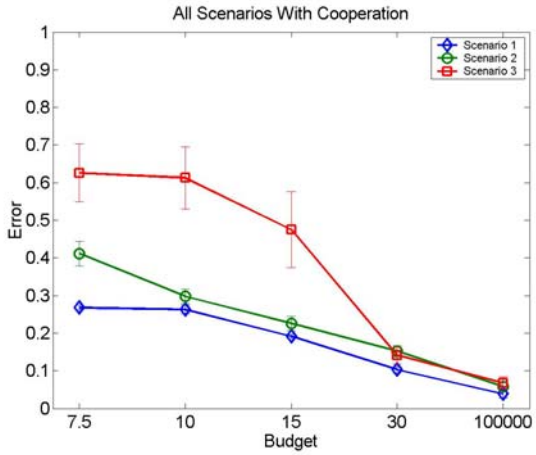

Figure 7: Performance with cooperation

Another interesting point is the S-shaped nature of the Scenario 3 curve in Fig. 7. This type of nonlinear change in behavior is typical of complex systems. In a realistic system the sharp drop is extremely useful in identifying "tipping points" in the system. In this case, for example, we can tell that increasing the budget from 10 to 15 has a much greater impact than, say increasing budget from 30 to 45. The ability to find tipping points is a key driver of success in the analysis of complex systems of this type.

Finally, we notice from Figs. 6 and 7 that the standard error increases as the complexity of the scenario increases and the budget limit increases. This probably reflects the fact that the GA needs more time to converge to similar solutions when the problem space is larger, which is natural and is expected. The complexity of the scenario and the budget limit are both factors that increase the search space. In the same vein, the standard error grows for non-cooperative simulations, which implies that the search for cooperative solutions converges faster.

4.6 Analysis of the evolved chromosomes

In this section we look at the actual values of the chromosome that were evolved in the experiments described above. The purpose of this analysis is to demonstrate the connection between the actual gene values (genotype) and the resulting behavior (phenotype), as well as to verify the interpretation of the chromosome representation.

4.6.1 Communications range

We first look at the gene for the communication range (radius) *r*. Each plot in Fig. 8 shows the results broken down by scenario, while cooperative and non-cooperative simulations are split into separate plots. The x-axis shows the increasing budget limit, while the y-axis has the average gene value for the particular case.

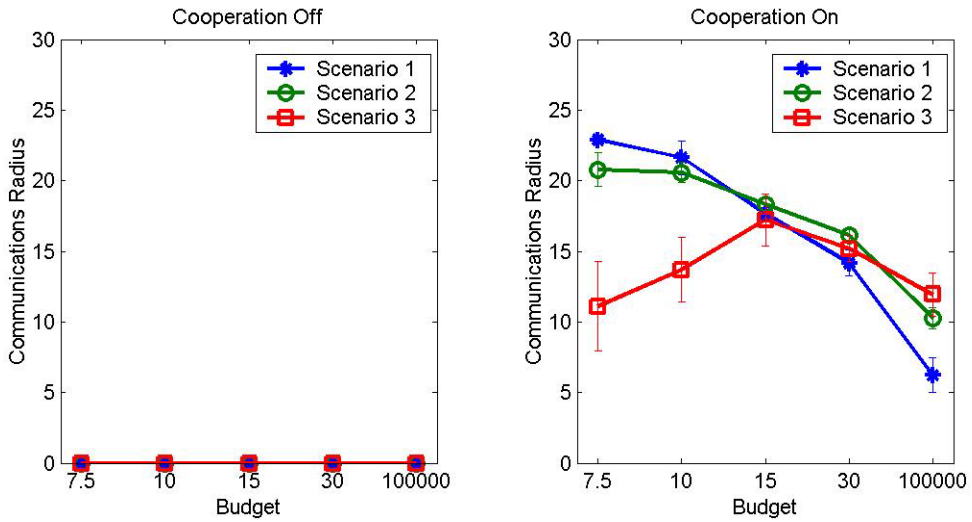

Figure 8: Communication radius gene values

Not surprisingly, for all runs where cooperation is disabled, the communications range is always zero. In the case where cooperation is enabled, however, we observe an interesting pattern: there is a decrease in the preferred communications radius with increasing budget in all cases except for the low budget cases under scenario 3. The decreasing trend is explained with the fact that as the budget increases, so does the ability to use more precise sensors or more numerous sensors. Hence, by allowing for greater sensor accuracy and density we need fewer neighbors to share information.

After closer inspection, we find that under scenario 3 at the lowest budget, three out of ten GA runs found a solution that sets the communication radius to zero. In other words, event though the GA has the option of using cooperation, in three of ten runs it found a solution that does not use cooperation. It would seem that the GA has found a local minimum similar to those discovered when cooperation is disabled. At the budget limit of 10, the GA used zero connectivity in only one out of ten runs. It is possible that running the GA for a larger number of generations might have allowed the GA to enforce cooperation in all ten runs under all budget levels. However, we did not test this in order to maintain consistency across all experiments, without requiring excessive computational time.

4.6.2 Sensor density and accuracy

Next, we look at the sensor density and accuracy genes. As mentioned earlier, there is one gene for the density of each available sensor type (sound, chemical, vibration). For space considerations we report here only the analysis of one sensor type (sound sensor). The results for other sensor types, which did not vary significantly from those reported here, will be reported in a later, full-length report.

As shown in Fig. 9, we find an increase in density with increasing budget, which is the expected behavior, since more money implies the ability to purchase more sensors. The pattern of increase, however, is different for the non-cooperative and cooperative systems. In the non-cooperative case (Fig. 9, left), there is a systematic decrease in sensor density as the complexity of the scenario increases. In contrast, in the cooperative case the three curves representing three difference scenarios do not differ significantly.

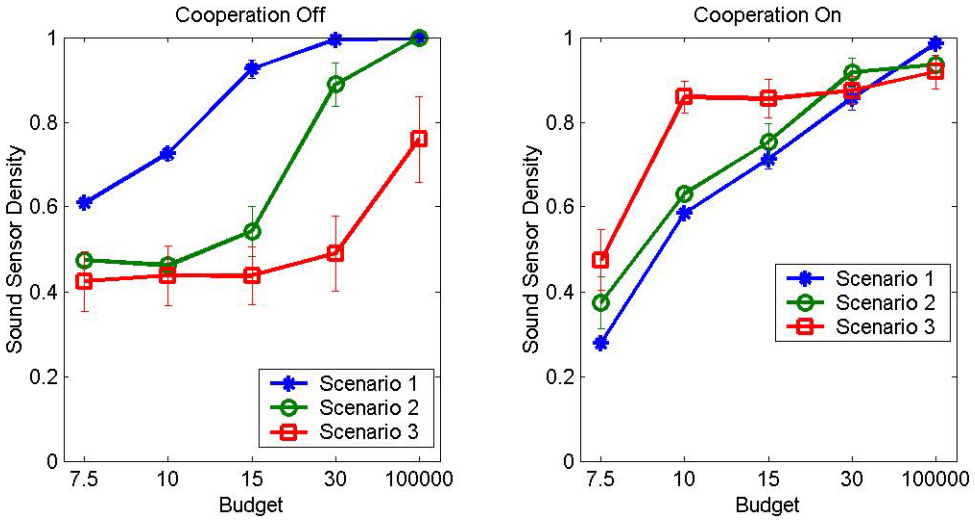

Figure 9: Sound sensor density gene values

Before trying to explain the strong dependence of sensor density on scenario complexity in the non-cooperative case, it is useful to examine the accuracy gene for the same conditions. The results are shown in Fig. 10.

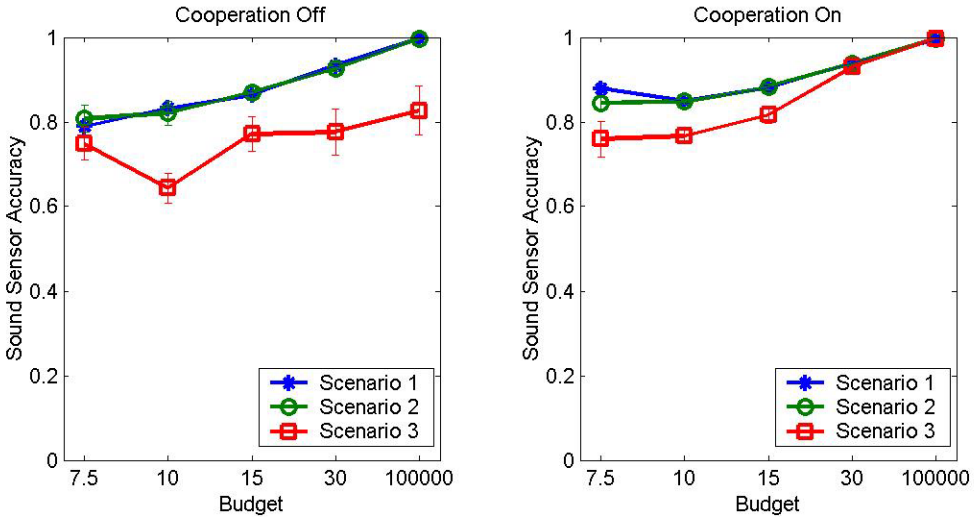

Figure 10: Sound sensor accuracy gene values

We see in Fig. 10 that the accuracy is always fairly high, with all but one of the points in excess of 70%. Because of the hyperbolic form of the sensitivity term in Eq. 1, the overall cost depends much more on accuracy than on sensitivity: even a slight increase in accuracy requires a dramatic decrease in density, an effect that is dramatically emphasized as accuracy approaches 100%. In the cooperative case, it is possible to decrease density without having much impact on performance, because smart dust units can collect information from their neighbors. In the non-cooperative case, increasing sensor accuracy is required because there are no neighbors to compensate for missing sensors – but this in turn requires further reductions in density.

By being able to maintain lower densities, cooperative systems can afford to have fewer, but more accurate sensors and this enables them to be effective even within lower budgets.

Finally, we have analyzed the genes that characterize the leaky integrators for the separate enemy types. As described earlier, there are two genes for each leaky integrator, which jointly characterize the dynamics of the response to sensory inputs. When we analyzed the combined effect of the two genes for each type of enemy, we saw fairly consistent values across all scenarios, so we do not report them here. The likely reason for this is that in the current study we presumed that all enemies move at the same speed, and that the signals they emit propagate to a fixed radius that is the same for all enemy types. Had we selected different speeds or signal propagation radii for different enemy types, it is likely that the GA would have evolved specialized sensors with different time constants.

5 Discussion and future work

The goal of this project was to contribute to a general understanding of network-centric systems and operations. We focused on the problem of designing a network-centric system in which we must simultaneously account for issues of resource allocation and system behavior and performance. While our results are preliminary, they already offer significant insights into network-centric operations.

Network-centric systems of the type described in this paper can be viewed simply as a specific instantiation of distributed systems. Some of the same theoretical analyses that apply to other types of complex systems can be applied to network-centric operations (e.g., Perry et al., 2002; Moffatt, 2003). We agree with these authors that developing a quantitative understanding of the performance of network-centric systems is going to be critical to their successful adoption in the military and in other contexts.

Our group has previously tackled projects of distributed computing and swarm intelligence that share a number of characteristics with the work described here. In one project, we developed distributed control algorithms for swarms of unmanned air vehicles performing search or SEAD missions (Gaudiano et al., 2003) and swarms of robots performing exploration and mapping functions (Rothermich et al., 2004). In all cases, two central tenets of our research have been (1) the development of decentralized algorithms for distributed, collaborative systems, and (2) the use of quantitative techniques including both computer simulation and mathematical analysis to understand and analyze the function of distributed systems.

The work we described makes novel inroads into our understanding of distributed command and control. We believe the most significant contribution of our work is that we are considering simultaneously two key issues that impact performance: the amount of available resources, and the algorithms used to perform distributed operations. Most of the work with which we are familiar tends to focus only on one of these problems. For instance, in the context of sensor networks, there are many algorithms to deploy sensors in an environment, to share information between sensors, or to control the amount of energy spent in performing operations. However, it is unclear how these results would generalize if additional constraints were taken into account.

Clearly, our own work is only embryonic in nature. Our initial experiments have raised more questions than they have answered. Nonetheless, we have already made some significant findings, first and foremost by showing that it is possible to take multiple types of constraints into account, and secondly by providing some quantitative results and analyses of the performance of networked systems. Among our more interesting results, we found that cooperation among networked units is beneficial, especially when resources are heavily constrained, or for scenarios of increasing complexity. While we are not the first to make this claim, we have found surprisingly little work that provides a quantitative verification of these claims often made for distributed C2 systems or network-centric systems.

We have already begun to extend the work presented here. For example, we recently developed an algorithm to modulate the complexity of the urban environment in which the sensor network is operating. These and other extensions will be the subject of future publications.

6 References

Alberts, D. S. and Hayes, R. E. (2003). Power to the Edge. Washington, DC: CCRP Publications.

Axelrod, R. (1997) The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration (Princeton University Press, Princeton, NJ).

Bonabeau, E. (2000) Business applications of agent-based simulation, Adv. Complex Syst. 3, 451-461

Bonabeau, E. (2002a) Agent-based modeling: methods and techniques for simulating human systems. *Proc. Nat. Acad. Sci. USA* 99, 7280-7287.

Bonabeau, E. (2002b) Predicting the Unpredictable. *Harvard Business Review*, 109-116.

Cebrowski, A. K. and Gartska, J. J. (1998). Network-centric warfare: its origin and future. *Proceedings* (a magazine of the Naval Institute), January 1998 issue.

Epstein J. M., Axtell R. L. (1996). Growing artificial societies: social science from the bottom up (MIT Press, Cambridge, MA).

Forrest, S. (1993). Genetic algorithms: Principles of adaptation applied to computation. Science 261: 872-878.

Gaudiano, P., Shargel, B., Bonabeau, E., Clough, B. (2003). Swarm Intelligence: a New C2 Paradigm with an Application to Control of Swarms of UAVs. *8th International Command and Control Research and Technology Symposium*. Washington, DC

Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing.

Moffatt, J. (2003). Complexity Theory and Network-Centric Warfare. Washington, DC: CCRP Publications.

Perry, W., Button, R. W., Bracken, J. Sullivan, T., Mitchell, J. (2002). Measures of Effectiveness for the Information-Age Navy: The Effects of Network-Centric Operations on Combat Outcomes. Washington, DC: RAND.

Rothermich, J. A., Ecemis, M. I. and Gaudiano, P. (2004). Distributed localization and mapping with a robotic swarm. In Sahin, E. and Spears, W. M. (Eds.) *Proceedings of the SAB 2004 International Workshop*, reprinted as Lecture Notes in Computer Science, v. 3342. New York: Springer-Verlag, 58-69.

7 Acknowledgments

We would like to thank CDR Gregory Glaros of the Office of Force Transformation for his extensive advice and guidance with this project and for his continued support of our work. We would also like to thank Gregory Cyr for his support through NUWC.